CS-Chord:基于聚类分离的分布式高维向量索引

袁鑫攀[1,2] 汪灿飞[1,2] 龙 军[3] 彭 成[1,2]
(湖南工业大学计算机学院 株洲 412007)[1]
(湖南工业大学智能信息感知及处理技术湖南省重点实验室 株洲 412007)[2]
(中南大学信息科学与工程学院 长沙 410083)[3]

摘 要 M-Chord是一种基于P2P网络的高维向量索引,其聚类边缘的向量容易与搜索圆频繁相交,使得查找的区域增多,降低了M-Chord的效率。提出一种基于聚类分离的分布式高维向量索引(CS-Chord),将边缘区域的高频检索向量从Chord环中分离出来,集中存储在服务器上,中心区域的向量仍存储于Chord环中,节省了大量资源的定位时间,从而提高检索效率。实验结果表明:在查询半径为0.2时,CS-Chord距离计算次数约为2000,比M-Chord减少了约2500次;CS-Chord消息转发次数约降低150次,仅为M-Chord的50%。

关键词 高维向量,聚类,Chord,分布式索引
中图法分类号 TP301.6 **文献标识码** A

CS-Chord:Distributed High Dimensional Vector Index Based on Clustering Separation

YUAN Xin-pan[1,2] WANG Can-fei[1,2] LONG Jun[3] PENG Cheng[1,2]
(School of Computer Science, Hunan University of Technology, Zhuzhou 412007, China)[1]
(Key Laboratory of Intelligent Information Perception and Processing Technology, Hunan University of Technology, Zhuzhou 412007, China)[2]
(School of Information Science and Engineering, Central South University, Changsha 410083, China)[3]

Abstract M-Chord is a high dimensional vector index based on the P2P network, and the sparse vector of the clustering edge is easy to intersect with the query circle, which makes the area of the query increased, and the efficiency of M-Chord is reduced. Based on the idea of M-Chord, this paper proposed a distributed high dimensional vector index (CS-Chord) based on clustering, which separates the edge vectors from the dense vectors. The vector of the central region is still stored in the Chord loop, which saves a lot of resources locating time and improves the efficiency of retrieval. Experimental results show that when the query radius is 0.2, the number of CS-Chord distance computation 2000, reducing about 2500 times compared with M-Chord. The message forwarding times of CS-Chord decreases about 150 times, only 50% of M-Chord.

Keywords High-dimensional vector, Cluster, Chord, Distributed index

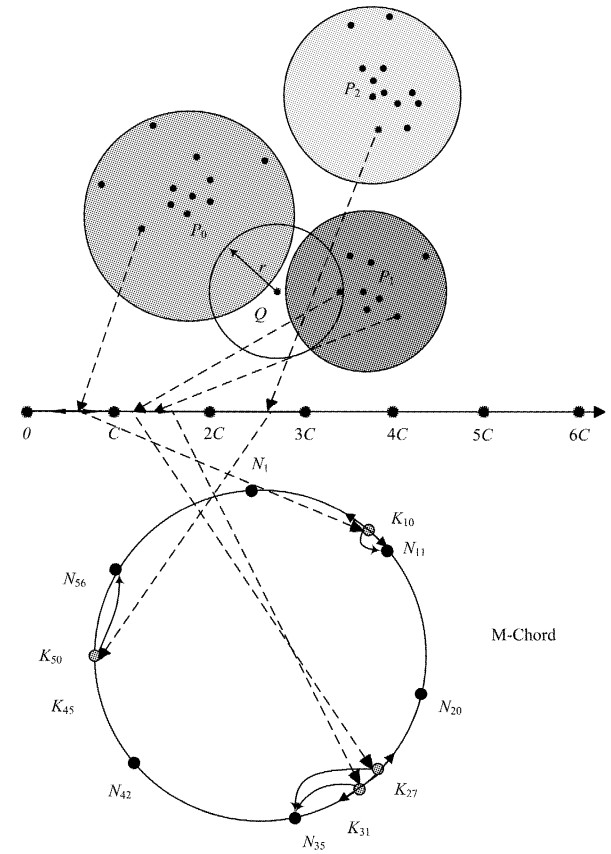
1 引言

基于内容而非属性检索的需求促进了相似性检索领域的发展,这种检索模型是由示例对象和所需查询对象的约束所定义的。将数据特征提取成高维向量,通过范围查询检索出目标对象。常见的索引结构有iDistance[1],KD-tree[2],R-tree[3],M-tree[4]等。随着数据量的日益上升,分布式处理的需求越发明显,目前主流的分布式索引结构有M-Chord[5],DKDT[6],R-Chord[7],MCAN[8]等。M-Chord将iDistance算法与Chord[9]协议相结合,通过iDistance将高维向量映射为一维关键值,采用Chord协议实现一维关键值的分布式存储与检索,从而降低了距离计算和消息转发次数。这种并行化计算和分布式存储使得M-Chord在检索领域有着广泛的应用。

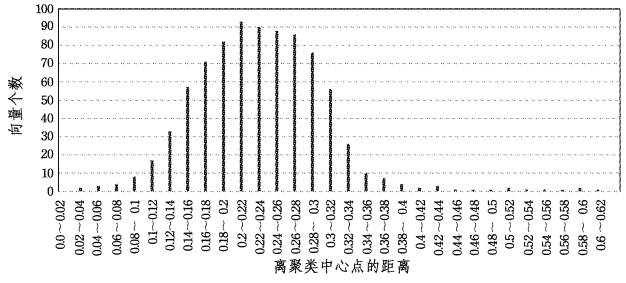
如图1(a)所示,M-Chord的聚类边缘向量一般比较稀少,这些稀少的向量使得每个聚类的半径变得很大。在范围查询时,半径越大越容易与范围查寻的搜索圆相交。只要搜索圆与聚类相交,相交区域的向量都必须在Chord环中定位。这些极少的边缘向量在Chord环中定位的次数相当频繁,降低了M-Chord的效率。为验证上述M-Chord存在的效率问题,本文采用K-means方法,对科雷尔图片特征数据(来自UCI数据集:http://archive.ics.uci.edu/ml/machine-learning-databases/CorelFeatures-mld/CorelFeatures.data.html)进行聚类,某聚类子空间向量分布图如图1(b)所示。该聚类的半径长度为0.62,绝大部分向量分布在0.1~0.34之间,但极少的边缘向量导致聚类的半径增加了将近一倍。图1(c)为在随机的1000个范围查找下该聚类空间中的向量检索频次图。由于范围查询时,搜索圆总是优先与聚类的边缘区域相交,边缘区域的向量虽然稀少,但被检索的频次较高,因此本文将边缘区域的高频检索向量从Chord环中分离出来,集中存储在服务器上,以省去大量资源定位的时间,从而提高检索效率。

本文受国家自然科学基金项目(61402165),湖南省自然科学基金项目(2015JJ3058),湖南省重点研发计划(2016JC2018),国家基金应急管理项目(S1651002),2016年湖南工业大学研究生校级创新基金项目(CX1606)资助。

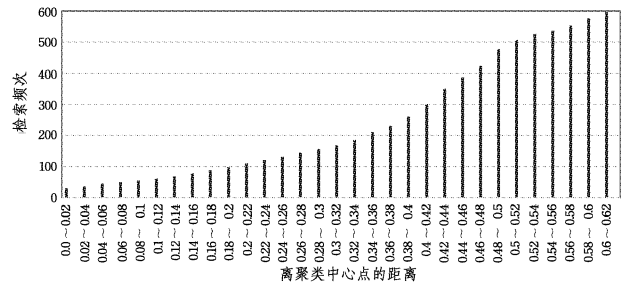
袁鑫攀(1982-),男,博士生,讲师,主要研究方向为信息检索;汪灿飞 男,硕士生,主要研究方向为信息检索;龙 军(1972-),男,教授,博士生导师,主要研究方向为网络资源管理与可信评估,E-mail:jlong@csu.edu.cn(通信作者)。



(a)检索示意图



(b)向量分布图



(c)向量检索频次图

图 1　M-Chord 图例

2　CS-Chord 分布式索引

首先将高维向量聚类，划分边缘区域和中心区域，再通过 iDistance 算法将高维向量转化为一维关键值 *iDist*，根据 *iDist* 的大小选择性地将向量存储在 Chord 环中或服务器上，并建立索引。

2.1　边缘向量分离

边缘区向量稀少且被高频检索，应该集中式存储，以省去 Chord 环中定位的时间。为达到这个目的，首先将聚类边缘的稀少向量分离出来。设聚类的边缘区向量与中心区向量的分界点为 R_b，聚类的半径为 R，则到聚类中心的距离$[0,R_b]$之间的区域为向量中心区，$[R_b,R]$之间的区域为向量边缘区。以图 2 为例，将二维空间的向量划分为 3 个聚类空间，聚类的深色部分为向量中心区，浅色部分为边缘区。中心区域的向量分布在$[0,3C]$之间，边缘区域的向量分布在$[3C,6C]$之间。

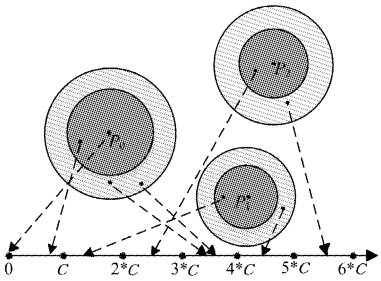


图 2　二维空间边缘向量分离示意图

假设向量 S 为一个需要加入 Chord 环的向量，P 为向量 S 所在聚类的中心点(锚点)，则向量 S 的关键值 $iDist(S)$ 的计算公式如下：

$$iDist(S)=\begin{cases} i*C+dist(S,P), & dist(S,P)\leqslant R_b \\ (i+n)*C+dist(S,P), & dist(S,P)>R_b \end{cases} \quad (1)$$

其中，$0\leqslant i<n$，n 是聚类空间个数，C 是常量，其值大于所有聚类半径。

通过式(1)可以将边缘的稀少向量分离，并且放入到一段连续的区域中。但是，由于稀少向量的检索频次高，因此使用独立的服务器集中存储稀少向量。这样当查询范围的 $iDist(S)\geqslant n*C$ 时，直接访问集中存储的服务器查询。

2.2　CS-Chord 索引结构

CS-Chord 是一种 P2P 网络中的高维向量索引。在 P2P 网络中，每台机器是对等的地位，因此在 Chord 环中的每个节点和服务器都应该具有同样的数据结构。这些节点都可以向 Chord 环中插入、删除以及查询数据，因此每个节点除了要包含基本的 Chord 环的一些信息外，还需要包含高维向量映射的一维关键值，具体如下：

(1)当前节点的标识符 ID、指针表(finger table)；

(2)当前节点的前一个节点的地址、后一个节点的地址以及服务器的地址；

(3)B^+-Tree 存储高维向量的一维关键值信息。

2.3　建立索引步骤

为一个向量 S 建立 CS-Chord 索引需要进行以下步骤(见图 3)。

步骤 1　计算向量 S 的关键值 $iDist(S)$。如果 $iDist(S)\geqslant n*C$，则将关键值 $iDist(S)$ 和向量 S 的信息发送到独立的服务器上，然后将其插入到服务器的 B^+-Tree 索引中；如果 $iDist(S)<n*C$，则转向步骤 2。

步骤 2　通过位置保持哈希函数 h 对 $iDist(S)$ 进行哈希，生成分配到 Chord 环上的标识符 $h(iDist)$。利用 Chord 定位算法查找标识符 $h(iDist)$ 应该存储的节点 IP。将向量的信息发送到该节点上，然后插入到该节点向量的 B^+-Tree 索引中，索引建立完成。

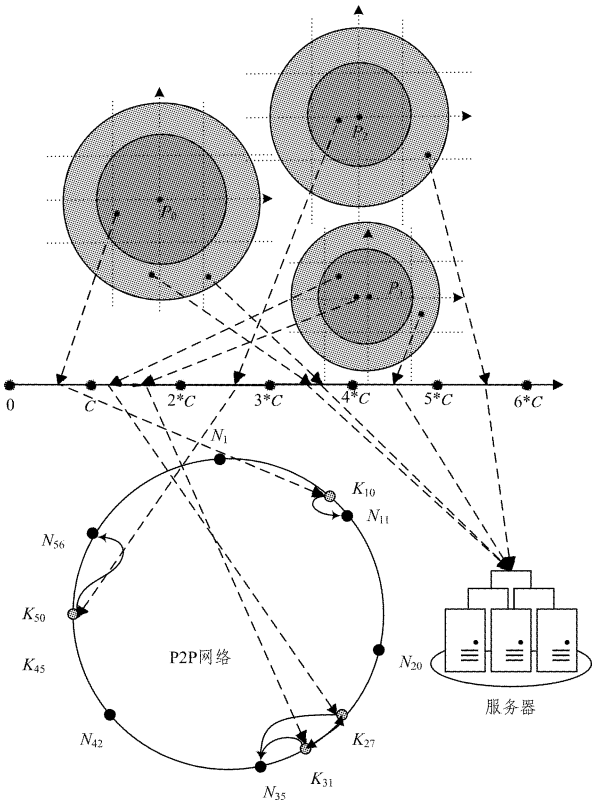


图 3 CS-Chord 索引示意图

位置保持哈希函数 h 的定义如下。

对于向量区间$[X_{\min},X_{\max}]$，要将其映射到区间$[Y_{\min},Y_{\max}]$上，且保持向量的一致性。假设 $X_i\in[X_{\min},X_{\max}]$，通过函数 h 映射后的值为 Y_i，$Y_i\in[Y_{\min},Y_{\max}]$。则该哈希函数可以定义为：

$$h(X_i)=Y_{\min}+\frac{(Y_{\max}-Y_{\min})*(X_i-X_{\min})}{X_{\max}-X_{\min}} \tag{2}$$

其中，因为 CS-Chord 的索引关键值的区间为$[0,K_{\max}]$，Chord 环的标识符空间范围为$[0,2^m-1]$，所以可将 $X_{\min}=0$，$X_{\max}=K_{\max}$，$Y_{\min}=0$，$Y_{\max}=2^m-1$ 代入式(2)，可得：

$$h(K_i)=\frac{(2^m-1)*K_i}{K_{\max}} \tag{3}$$

需要说明的是，位置保持哈希函数 h 的存在原因在于：通常的映射方法先对向量的关键值哈希，然后再对 2^m 取模，得到的值即为 Chord 环的关键值。但是这种做法将原本相邻的向量映射到了不同的节点上。这对于 iDistance 这种需要连续查找的索引算法来说是不可取的，因此需要一种位置保持的哈希函数，以保持向量顺序的一致性。

3 CS-Chord 的范围查询算法

图 4 给出了 CS-Chord 的范围查询 $Range(q,r)$ 的示意图，其中 q 为待查向量，r 为查询范围半径，具体步骤如下：

(1)通过 iDistance 计算出范围查询 $Range(q,r)$ 与聚类的相交区域，映射为多个关键值区间$[x_i,y_i]$。

(2)如果 $x_i<n*C$，则转向步骤(3)；如果 $x_i\geqslant n*C$，则将步骤(1)计算的信息发送到独立服务器上，转步骤(4)。

(3)生成 Chord 环中的关键值范围$[h(x_i),h(y_i)]$。查询路由表定位标识符 $h(x_i)$ 所在的节点，如果 $h(y_i)$ 大于节点中所存向量的标识符最大值 $h(iDist)_{\max}$，则将范围$[h(iDist)_{\max},h(y_i)]$发送到该节点的后继节点；如果 $h(y_i)$ 比后继节点的 $h(iDist)_{max}$ 大，则继续向它的后继节点发送查询信息。

(4)每一个节点(包括服务器和 Chord 环中的节点)接收到查询请求，在此节点的 B^+-Tree 中检索关键值范围中是否有向量存在，若存在向量则与待查向量 q 进行距离计算，若距离小于 r，则返回到最初发送请求的节点。

如图 4 所示，该查询 q 与聚类 P_0 中心区域、边缘区域都相交，与聚类 P_1 边缘区相交。映射的一维关键值范围为$[x_1,y_1]$，$[x_2,y_2]$，$[x_3,y_3]$。$[x_1,y_1]$区间送往 Chord 环中检索，$[x_2,y_2]$和$[x_3,y_3]$区间送往服务器中检索。

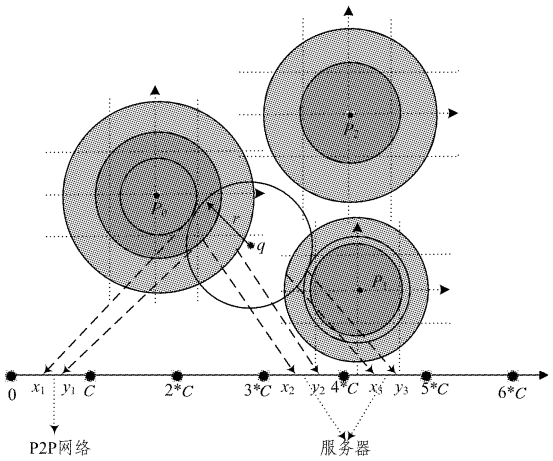


图 4 CS-Chord 范围查找示意图

4 实验结果与分析

实验采用 Peersim 模拟器来模拟 P2P 网络进行测试，Chord 协议由 Peersim 官网提供，设置 Chord 环节点数为 1000。实验数据采用科雷尔图片特征集，包括 68040 幅图像，32 维颜色特征。采用 K-means 方法聚类，初始聚类数为 64 个。统计了 1000 个随机查询向量在不同查询半径下的平均距离计算次数、消息转发次数、运行时间以及返回的候选集占整个数据集的百分比。实验平台为：Windows 7 旗舰版 sp1 64 位；硬件配置为：CPU 4 核 2.5GHz，8GB 内存，500G 硬盘；程序开发语言为 Java。

4.1 距离计算次数

如图 5 所示，实验统计了不同查询半径下，DKDT，M-Chord，R-Chord，CS-Chord 4 种索引结构的平均距离计算个数。

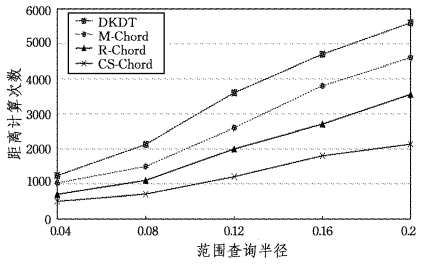


图 5 不同半径下的距离计算次数

由于 DKDT 是基于 KD-tree 划分节点的，导致很多节点包含很少的向量，向量之间距离计算成本增加。M-Chord 中 iDistance 算法将高维向量映射到一维轴上，有效避免了大量

的距离计算。R-Chord 与 M-Chord 类似，通过增加相对位置码、缩小搜索区间来减少距离计算。CS-Chord 将高频检索向量存储在服务器上，缩小了搜索区间，进一步减少了距离计算。

4.2 消息转发次数

如图 6 所示，实验统计了不同查询半径下，DKDT，M-Chord，R-Chord，CS-Chord 4 种索引结构的平均消息转发的次数。在 DKDT 索引结构中，接收到查询的节点不仅要搜索本地数据库，还要根据其 TIB(局部数据库)将查询转发到其他节点进一步搜索，消息转发次数较高。M-Chord 和 R-Chord 每个节点只需负责一个查询区间，消息转发次数相近。CS-Chord 算法将边缘高频检索向量集中存储，在范围查询时，这部分向量可以直接在服务器中快速查询，避免了在 Chord 环中的定位操作，因此 CS-Chord 的消息转发次数小于其他 3 种算法。

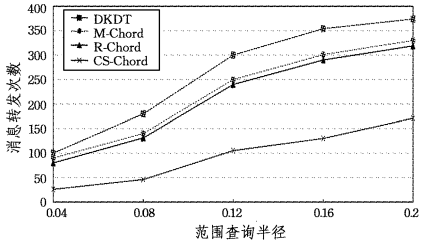


图 6　不同查询半径下的消息转发次数

4.3 运行时间

如图 7 所示，实验测量了不同查询半径下，DKDT，M-Chord，R-Chord，CS-Chord 4 种索引结构的平均运行时间。图 7 表明，CS-Chord 的运行时间远小于其他 3 种索引结构。这种运行时间上的优势主要是距离计算次数和消息转发次数减少带来的。

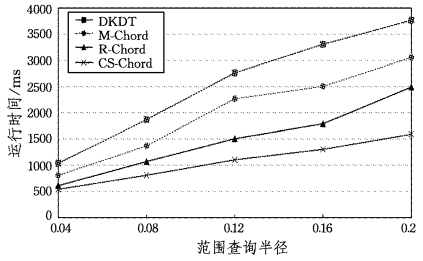


图 7　不同查询半径下的运行时间

4.4 返回的候选集

如图 8 所示，在不同查询半径下，不同索引返回的候选集随着查询半径的增大而增加。因为随着查询半径的增大，查询范围与聚类相交的部分也越来越大，各索引返回的候选集也越来越大。DKDT 返回的候选集最多，CS-Chord 和 M-Chord 方法返回的候选集基本一致，而 CS-Chord 将高频检索向量存储在服务器上，需要的时间更少。R-Chord 通过相对位置码缩小搜索区间，从而返回了更少的候选集，但需要更多的空间和时间来进行位置码比较，以缩小候选集。

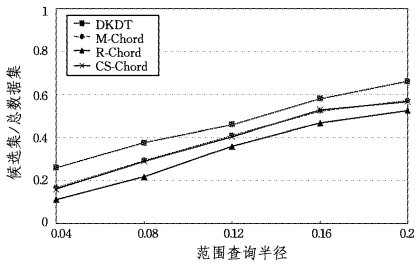


图 8　不同查询半径下返回的候选集

结束语　本文提出一种 CS-Chord 分布式索引，将边缘区域向量分离，并集中存储在独立的服务器上，而中心区域的向量仍存储在 Chord 环中。高频的查询集中在独立服务器，减少了 Chord 环上的搜索范围，从而提高了检索效率。依据实验得出以下结论：

(1)距离计算次数实验表明，CS-Chord 的距离计算次数约为 M-Chord 的一半，远小于其他几种索引结构。

(2)消息转发次数实验表明，CS-Chord 的存储方式使得其在范围查询时频繁的定位操作集中在服务器，消息转发次数在这 4 类索引中最少。

参 考 文 献

[1] SCHUH M A, WYLIE T, BANDA J M, et al. A comprehensive study of iDistance partitioning strategies for knn queries and high-dimensional data indexing[M]// Big Data. Springer Berlin Heidelberg, 2013: 238-252.

[2] ZHOU K, HOU Q, WANG R, et al. Real-time KD-tree construction on graphics hardware [J]. ACM Transactions on Graphics, 2008, 27(5): 126.

[3] ARGE L, BERG M D, HAVERKORT H. The priority R-tree: A practically efficient and worst-case optimal R-tree [J]. Acm Transactions on Algorithms, 2008, 4(1): 1-30.

[4] LIU X. The Dirac Operator on Regular Metric Trees[J]. Physics, 2015, 226: 165-178.

[5] NOVAK D, ZEZULA P. M-Chord: a scalable distributed similarity search structure[C]// International Conference on Scalable Information Systems (Infoscale 2006). Hong Kong, 2006: 1-10.

[6] ALY M, MUNICH M, PERONA P. Distributed Kd-Trees for Ultra Large Scale Object Recognition[C]// British Machine Vision Conference. 2011: 1-11.

[7] YIN W, ZHU M, JIANG L. R-Chord: A distributed similarity retrieval system with RPCID[C]// IEEE International Conference on Network Infrastructure and Digital Content, 2009. IcNidc, 2009: 393-399.

[8] 吴纯青，任沛阁，王小峰. 基于语义的网络大数据组织与搜索[J]. 计算机学报，2015，38(1)：1-17.

[9] STOICA I, MORRIS R, KARGER D, et al. Chord: A scalable peer-to-peer lookup service for internet applications[J]. IEEE/ACM Transactions on Networking, 2003, 11(4): 149-160.